Hierarchical Routing Using Link Vectors

Jochen Behrens
Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, California 94303
Jochen.Behrens@eng.sun.com

J.J. Garcia-Luna-Aceves
Computer Engineering Department
School of Engineering
University of California
Santa Cruz, California 95064
jj@cse.ucsc.edu

Abstract—
An area-based link-vector algorithm (ALVA) is introduced for the distributed maintenance of routing information in very large internetworks. According to ALVA, destinations in an internetwork are aggregated in areas in multiple levels of hierarchy. Routers maintain a database that contains a subset of the topology at each level of the hierarchy. This subset corresponds to those links used in preferred paths to reach destinations (nodes inside the same immediate area or remote areas). ALVA is the first hierarchical routing algorithm based on link-state information that does not require complete topology information at each level in the hierarchy. The correctness of ALVA is verified. Simulation results are presented showing that ALVA outperformes OSPF in terms of communication and storage overhead.

I. Introduction

In the past, most work in distributed routing has proceeded in two directions: protocols based on *distance-vector algorithms* (DVA) and protocols based on *link-state algorithms* (LSA). Most distance-vector protocols are based on a distributed implementation of the Bellman-Ford algorithm to compute shortest paths [1]. Link-state algorithms, on the other hand, are based on the flooding of link information; they require the complete topology information to be replicated at every node [2], [3], [4]. Recently, we introduced *link-vector algorithms* (LVA) [5] to address the scaling problems associated with traditional DVAs and LSAs.

Although we have shown that LVAs are more scalable than LSAs and DVAs, using LVAs with a flat addressing structure is not sufficient for a net to scale to very large numbers of nodes and destinations. Any routing algorithm that requires routers to know about every single destination in an internet, becomes infeasible as the internet grows. The storage requirements as well as computational and communication overhead become too costly. To address this problem, the amount of information stored and communicated must be reduced using address aggregation schemes.

The goal of any address aggregation scheme is to reduce the size of the topology databases or routing tables kept at routers, thereby reducing the amount of data that needs to be communicated, processed, and stored. The main idea in aggregation schemes is that a router keeps in its database one entry per node or link that is "close," and an entry for a set of nodes or links further away [6]. To achieve this, hierarchies of addresses are formed by grouping together ("clustering") nodes that are close together.

The OSPF [4] and ISO IS-IS [2] protocols define areas that correspond to well defined portions of an internet. Areas are defined statically, and to route traffic among such areas, a backbone is used to interconnect all areas. In OSPF, all inter-area traffic must be routed via the backbone.

There have been many hierarchical routing proposals described in the past based on the notion of areas, which are also called clusters [7]. The first such proposal was McQuillian's [8]; this proposal was analyzed in detail by Kamoun and Kleinrock [9]. Most prior proposals on hierarchical routing have routing algorithms based on topology broadcast or variations of the distributed Bellman-Ford algorithm (e.g., [10], [11]). More recently, Murthy and Garcia-Luna-Aceves [12] proposed an area-based hierarchical routing algorithm called HIPR that is based on McQuillian's clustering scheme and the loop-free path finding algorithm [13] which is a loop-free algorithm based on distance vectors. Ramamoorthy et al. [14], [15] proposed an algorithm based on link-state information for hierarchical routing. According to this algorithm, a node maintains complete topology information of each area to which the node belongs, and the topology of an area at a given level is given by the interconnection of the lower-level areas within it.

We introduce a new area-based hierarchical routing scheme that uses LVA as its basic routing algorithm. This new scheme, which we call area-based link-vector algorithm (ALVA) supports multiple levels of hierarchy and does not rely on a backbone for inter-area routing. ALVA allows more flexible topologies and shows improved performance by removing the bottleneck backbone. The main motivation for this new scheme is to provide an approach based on link-state information that does not require complete topology information for each hierarchical level. As we show subsequently, it constitutes the basis for developing internet routing protocols based on link-state information that are much more scalable than OSPF. The next section describes the network model and gives a short overview over LVA. Section III describes the hierarchical routing algorithm. Section IV proves its correctness. Section V discusses its complexity and presents simulation results addressing its average performance.

II. Background and Network Model

A. Network Model

An internet is modeled as an undirected, weighted graph $G = (V, E)$, where V is the set of nodes (routers) and E is the set of edges (links). Each point-to- point link has two costs associated with it – one for each direction. (If multiple routing policies are used, multiple costs can be assigned in each direction. However, for a given policy, the cost must be unique).

This work was supported in part by the Advanced Research Projects Agency (ARPA) under contract F19628-96-C-0038

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1998		**00-00-1998 to 00-00-1998**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Hierarchical Routing Using Link Vectors	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**9**	

Nodes of the graph are clustered into subgraphs called areas. Although the new hierarchical routing algorithm can be used with overlapping clusters with only minor modifications, for simplicity, we assume that the areas are disjoint, i.e. every node belongs to exactly one area.

An underlying protocol assures that

- A node detects within finite amount of time the existence of a new neighbor or the loss of connectivity with a neighbor.
- All messages transmitted over an operational link are received correctly and in proper sequence within a finite amount of time.
- All messages, changes in cost of a link, link failures, and new-neighbor notifications are processed one at a time and in the order in which they are detected.

Each router and each area has a unique identifier, and any other router can determine from that identifier to which area the router belongs. Link costs may vary in time, but are always positive. Furthermore, routers are assumed to operate correctly, and information is assumed to be stored without error.

B. Link Vector Algorithm

LVA [5] is based on the dissemination of partial topology information. Routers propagate incremental information only about those links that they actually use to reach any destination. Thus, all routers keep a partial topology. A local *path selection algorithm* is used to compute their *source graph* based on that partial topology. For example, in shortest-path routing, the path selection algorithm could be Dijkstra's algorithm and the source graph is the shortest-path tree. However, the source graph need not be a tree, it can contain multiple paths for the same destinations to support multipath routing and it can contain links used by different routing policies.

Because routers in LVA have differing topology databases, it is important that the information is consistent to avoid the possibility of long-term or permanent looping. To achieve this, routers tell their neighbors which links they use and which links they no longer use, using *add* and *delete* updates. An update specifies all the parameters of the link and a router sends an update in a message only when a link is modified, added, or deleted in its source graph. This way, the source graph is reported to neighbors incrementally, and a typical control message contains only a few link-state updates. In addition to the link parameters, a router must record the set of neighbors that reported the link, a link that is not reported by any neighbor must be removed from the topology database, unless it is a link originating at the router itself.

III. AREA-BASED LVA (ALVA)

According to ALVA, nodes are clustered into areas organized into multiple hierarchical levels, so that areas can be grouped into higher-level areas as well. Figure 1 shows an example topology with three levels of hierarchy. Links in this topology are assumed to be bidirectional, with unit cost in both directions. The nodes (named in lower case) make up level 0 in the hierarchy. Level 1 consists of the areas A1..A5, B1..B3, and C1..C4, while we have the areas A, B, and C at the top level, level 2. In this example topology, only border-nodes are named, with the exception of node x, which is an interior node of area $A4$.

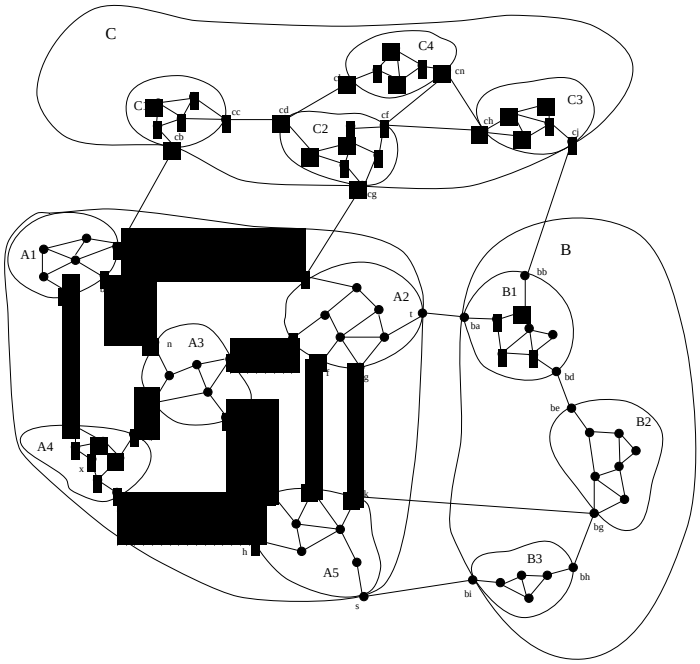

Fig. 1. 2-level hierarchical network

A *border node* is a node that has a link to a node that belongs to a different area. A k-level border node is a node that connects k-level areas. Nodes can determine to which area a given address belongs, and at which level of the hierarchy two given addresses differ. With this information, nodes can dynamically determine whether they are border nodes (this may change with link failures or establishments,) and at which level their border is. The basic operation of ALVA is as follows:

- For routing within an area, flat LVA is used.
- For inter-area routing, LVA is applied on the topology representing the connectivity among areas at any particular level.

At any given level, shortest-path routing is used among all areas that are contained in the same area one level up in the hierarchy. Because areas are seen as single entities by remote routers, the cost to traverse them cannot easily be determined. Given that the cost of the links between the areas is outweighed by the area traversal cost, using the actual for those links need not improve overall performance of the algorithm. Accordingly, for simplicity, we use minimum hop routing across areas in this paper.

Figures 2, 3, and 4 provide a formal specification of ALVA. The following sections are used to describe the information stored and communicated, as well as ALVA's operation, in more detail. For simplicity, we assume that the sequence numbers used to validate updates are based on unbounded counters. In practice, a mechanism using a finite sequence number space must be used.

A. Information Maintained at Nodes

With respect to the information exchanged, all routers act as peers in ALVA. This does not mean that the information stored is the same at all routers, but that the type of information is the same. There are no special routers that need to store any additional information. Thus, we can ensure that any routers can, without delay, accept the additional functionality of a border router if a

```
procedure update (i, message)
    update_topology_table (i, message)
    if updated then
        build_source_graph
        build routing table
        compare_source_graphs (i)
        ST_i = NewST_i
end update

procedure update_topology_table (i, message)
    for all updates in message do
        if local_address ( j ) then
            update as in plain LVA, using table TT_i^0
        else  – – remote address at level l (implies that k is
              – – remote address as well)
        if type = ADD then
            if (j,k) ∉ TT_i^l then
                add link to TT_i^l
            else if h is head of link then
                if sn > TT_i^l(j,k).h.sn then
                    change link values
                else if sn = TT_i^l(j,k).h.sn then
                    add reporting node n
                else if sn < TT_i^l(j,k).h.sn then
                    send correction to n
            else
                add h as head of link
        else – – type = DELETE
            if h is head of link then
                if (j,k) ∈ TT_i^l then
                    if sn > TT_i^l(j,k).h.sn then
                        mark h as deleted
                        if no other head
                            mark link as deleted
                    else if sn = TT_i^l(j,k).h.sn then
                        delete reporting node n
                        if no reporting node for h then
                            mark h as deleted
                            if no other head then
                                mark link as deleted
                    else if sn < TT_i^l(j,k).h.sn then
                        send correction to n
                else
                    TT_i^l(j,k).h.sn = sn
end update_topology_table
```

```
procedure link_change ( i, j )
    if local_address (j) then
        TT_i^0(i,j) = (c_i^j, new_sn, {i})
    else
        TT_i^0(i, area(j)) = (c_i^j, new_sn, {i})
    build_source_graph
    build routing table
    compare_source_graphs (i)
    ST_i = NewST_i
end link_change

procedure link_up ( i, j )
    N_i = N_i ∪ j
    if local_address (j) then
        TT_i^0(i,j) = (c_i^j, new_sn, {i})
    else
        TT_i^0(i, area(j)) = (c_i^j, new_sn, {i})
    build_source_graph
    build routing table
    compare_source_graphs (i)
    ST_i = NewST_i
end link_up

procedure link_down ( i, j )
    N_i = N_i − j
    for all (k,l) ∈ TT_i do
        TT_i^0(k,l).r = TT_i^0(k,l).r − j
        if TT_i^0(k,l).r = ∅ then
            mark (k,l) as deleted
    if local_address (j) then
        TT_i^0(i,j) = (c_i^j, new_sn, {i})
    else
        TT_i^0(i, area(j)) = (c_i^j, new_sn, {i})
    build_source_graph
    build routing table
    compare_source_graphs (i)
    ST_i = NewST_i
end link_down
```

Fig. 3. ALVA specification

update: tuple $(j, k, c_j^k, sn, type, [h])$
- j, k: origin and destination of link
- c_j^k: cost of link (j, k)
- sn: sequence number
- $type$: ADD or DELETE
- h: head of link, if origin is area address

topology table TT_i at node i with entries:
- j, k: origin and destination of link
- if j local address in same area:
  - c_j^k, sn: cost and sequence number
  - list of reporting node
- if j area address
  - c_j^k: connectivity info
  - for each reported head of link
    - sn: sequence number
    - list of reporting nodes
  - if more than one head in list, indicate which one forwarded

TT_i can be subdivided into TT_i^l, where l indicates the level in the hierarchy.

source graph ST_i, new source graph $NewST_i$

Fig. 2. Variables and Data Structures for ALVA specification

new link crossing a border is established.

Each router maintains a topology table and a source graph. The latter is used to derive the routing table. The topology table may be viewed as being split up into one table for each level in the routing hierarchy (as is assumed in the pseudo code in Figures 3 and 4); this is merely an implementation matter for the path-selection algorithm.

In principle, the topology table contains the following information about all links known to the router, and belonging to the router's own 1-level area: the head and destination of the link, the cost of the link, the link's sequence number, and the list of the reporting nodes for the link. Again, the reporting nodes of a known link are those neighbors of the router who have reported using that link. If more than one routing policy is used in the network, multiple costs can be reported for the same link.

For inter-area links, additional information must be stored. Because an inter-area link represents connectivity rather than a particular physical link, it may be that this link actually corresponds to multiple links. Thus, checking whether an update concerning such a link is recent becomes a problem, given that there can be no unique sequence number assigned to it. To solve this problem, the sequence number and the ID of the head of the actual link are stored. Different neighbors may report different heads about the same inter area connections to a node. The node then stores all the different heads concerning the connection, but forwards only one of them. To reduce communication overhead, all nodes should use the same criterion as to which head to report in such a case.

Of course, the list of reporting nodes must be kept on a per

```
procedure compare_source_graphs (i)
    for all (j,k) ∈ NewST_i, (j,k) ∉ ST_i or NewST_i(j,k).sn > ST_i(j,k).sn or change in head of link do
        check_link_in_source_graphs (i, j, k, ADD)
    for all (j,k) ∈ ST_i, (j,k) ∉ NewST_i do
        check_link_in_source_graphs (i, j, k, DELETE)
    if border-node (i) then
        send (inter_area_message)
    send (intra_area_message)
end compare_source_graphs

procedure check_link_in_source_graphs (i, j, k, type)
        if border-node (i) then
            if i = j then
                if not local_address (k) then
                    inter_area_message = inter_area_message ∪ (area(i),k,connectivity,sn,type,i)
                intra_area_message = intra_area_message ∪ (j, k, c_j^k, sn,type)
            else
                if remote_address (j) (level l) then
                    inter_area_message = inter_area_message ∪ (j, k,connectivity,TT_i^l(j,k).h.sn,type,TT_i^l(j,k).h)
                    intra_area_message = intra_area_message ∪ (j, k,connectivity,TT_i^l(j,k).h.sn,type,TT_i^l(j,k).h)
                    if change in h and type = ADD then
                        inter_area_message = inter_area_message ∪ (j, k,connectivity,TT_i^l(j,k).h.sn,DELETE,TT_i^l(j,k).h)
                        intra_area_message = intra_area_message ∪ (j, k,connectivity,TT_i^l(j,k).h.sn,DELETE,TT_i^l(j,k).h)
                else
                    intra_area_message = intra_area_message ∪ (j, k,c_j^k,TT_i^l(j,k).sn,type)
                    if remote_address (k) (level l)then
                        inter_area_message = inter_area_message ∪ (j, k,connectivity,TT_i^l(j,k).h.sn,type,j)
        else – – interior node
            if remote_address (j) (level l)then
                intra_area_message = intra_area_message ∪ (j, k,connectivity,TT_i^l(j,k).h.sn,type,TT_i^l(j,k).h)
                if change in h and type = ADD then
                    intra_area_message = intra_area_message ∪ (j, k,connectivity,TT_i^l(j,k).h.sn,DELETE,TT_i^l(j,k).h)
            else
                intra_area_message = intra_area_message ∪ (j, k,c_j^k,TT_i^l(j,k).sn,type)
end check_link_in_source_graphs
```

Fig. 4. ALVA specification (*Cont.)*

```
level 0:
   all links for paths
     x -> r
     x -> q
     x -> p
     (and other paths
      within A4)

     p - A5
     q - A3
     r - A1
```

```
level 1:
    A1 - A2, c
    A3 - A2, m
    A5 - A2, j (, k)
    A2 - B,  t
    A5 - B,  s (, k)
    A1 - C,  c
    A2 - C,  d
  may store
    A4 - A1, r
    A4 - A3, q
    A4 - A5, p
```

```
 level 2:
    (none)
  may store
    A - B, s (, t)
    A - C, c (, d)
```

Fig. 5. Topology at Node x

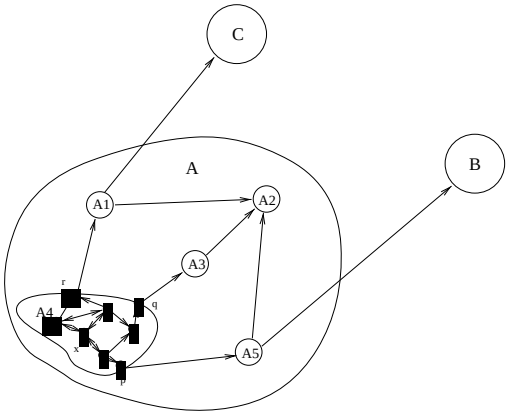

Fig. 6. Topology at Node x

head-of-link basis as well. The list of reporting nodes can easily be stored as a bit vector, because only neighbors of a node can be in that list. Thus, the storage overhead of that list is relatively minor.

The source graph contains all links that are used on a preferred path to any destination, as determined by the local path selection algorithm. In the case of shortest-path routing, it is simply the shortest-path tree.

Figure 5 shows a textual representation of the links known at node x. Figures 6 and 7 show a graphical representation of the topology databases at node x and the border node k. As can be seen in Figures 5 and 6, x knows all the links necessary for it (or one of its neighbors) to reach any destination within the level 1 area $A4$. In particular, it knows all the links necessary to reach the border-nodes. In addition, the local table contains links from these border-nodes to the neighboring level 1 areas. (The internal topology of $A4$ is too small to show any significant saving in space as compared to topology broadcast here. However, it should be noted that a few of the links are known only in one (the "useful") direction, exhibiting some of the savings due to LVA. At the next level of hierarchy, the figures show a partial view of the inter-area topology. Note that, while node x sees only one way to each of the level 2 areas B and C, border-node q in the same area actually knows about the alternatives through area $A2$, enabling it to react fast to changes in the topology and then propagate that information within its area.

Although the information concerning links leaving their own

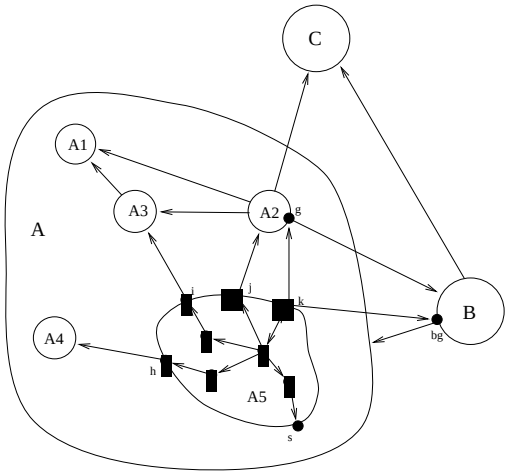

Fig. 7. Topology at Node k

area in levels 1 and 2 is redundant, it may be beneficial to store it in the tables to simplify the path selection algorithm.

Figure 7 shows the topology as seen by border-node k of area $A5$. It can be seen that the two topologies are quite different. However, due to the way the tables are formed, these differences cannot create any routing loops. Note that, by virtue of being a border-node to area B, k actually knows about the connection between areas B and C. It does not propagate this information within its area though, because it prefers the path through $A2$ to reach C.

B. Information Exchanged between Nodes

While there is no difference in the data stored at nodes, the information exchange within an area is obviously different from the inter-area exchange. Border nodes filter the information that is forwarded across their borders. They do not forward internal information about their area across the area border, but add the appropriate head of link information to updates concerning links leaving their area.

Whenever there is a change in a node's source graph, it sends incremental updates about the change to its neighbors: it sends an *add* update for links that they are using to get to any destination; it sends a *delete* update for links that they used before but that are not used any more. For links within an area, each such update contains the cost, the sequence number, and the type. Updates concerning links crossing area borders contain the areas of origin and destination, a sequence number and the head of the link reporting that sequence number.

For border links (i.e., links attaching to other areas), a border node distributes information concerning the link within its own area specifying the actual head of the link and the area address as the destination of the link. A border node makes sure that no link from within its area is reported to its peer in the other area. Links to other areas are converted to a hierarchical form and one of the actual heads of that connectivity is chosen to propagate the respective sequence number.

To illustrate the differences in how information is forwarded by border-nodes, Figure 8 shows which link states are forwarded by node o. Because o must distinguish between the recipients of its update packets, it assembles different packets to neighbor q and its interior neighbors. The differences are further illustrated in Figure 9, where the links that are forwarded by node h to its neighbor p, which is in a different area at level 1, and the links that s forwards to its neighbor bi over a level 2 boundary.

```
forwarded to q:          forwarded to neighbors in A3:
     A3 - A1, n               o  - A4
     A3 - A2, m               A1 - C,  c
     A3 - A5, l               A5 - B,  s
     A1 - C,  c               links used within A3
     A5 - B,  s
```

Fig. 8. Link-states forwarded by Node o

```
h forwards to p:          s forwards to bi:
     A5 - A2, j                A - B, s
     A5 - A3, i                A - C, d
     A5 - A4, h
     A5 - B,  s
     A2 - C,  d
```

Fig. 9. Link-states forwarded by Node h and s over area boundaries

C. Operation of ALVA

The operation of ALVA is very similar to that of basic LVA. When an update message is received from a neighbor, every update in the message is examined and the topology table changed as necessary.

First, consider that the update is an *add*: if there is no information about the link in the topology table, then the link is added to it. If the link is already present in the table, then its value is changed if the sequence number indicates a more recent update. If the sequence number is the same as the one stored, then the neighbor that sent the update message is added to the list of reporting nodes.

In the case of a *delete* update, the sender of the message is removed from the set of reporting nodes and, if the set becomes empty, the link is removed from the topology table.

In either case, an update containing recent information is sent back to the neighbor who sent the message if an update is found to be out of date (i.e., its sequence number is smaller than the one stored).

If there was any change in the topology table, then the updated topology table is used to obtain the new source graph, using the local path selection algorithm. The routing table is then updated from the new source graph. Finally, the new source graph is compared with the previous one to assemble the update packets that are sent to the neighbors. In principle, an *add* update is generated for any new link in the source graph, and for any link whose sequence number changed as a result of the update procedure. For any link that was previously part of the source graph but is no longer being used, a *delete* update is generated. Of course, a border node must filter the propagation of this information as described above.

The main difference between ALVA and LVA lies in the fact that the sequence numbers and reporting nodes for inter-area links are updated on a per head-of-link basis in ALVA. This also means that, if the head of the link changed for some inter-area link, two updates must be generated, one to delete the old head and another to add the new one. Because all routers use the same criterium to choose which head to advertise, this is a rare occurrence.

IV. CORRECTNESS OF ALVA

The proof of correctness for ALVA assumes that update messages are transmitted reliably and received and processes in the order that they are sent. In addition, we assume that there is a finite number of changes in link state up to time t_0, after which time there are no more changes. With these assumptions, the following theorem shows that ALVA is correct.

Theorem 1: After a finite time after t_0, no more updates are sent in the network, and all routers have up-to-date link-state information in their topology table and have computed correct hierarchical source graphs.

The proof of correctness consists of two parts: first, it is shown that ALVA terminates. The second part shows that the information in the network is consistent upon termination and thus correct routes have been computed. Both parts of the proof build on the properties proven for LVA [5] and extend the proofs for LVA to the hierarchical algorithm. The following lemmas constitute the proof.

Lemma 1: ALVA terminates within a finite amount of time after t_0.

Proof: The proof that the hierarchical LVA terminates is by induction on the number of levels (k) in the hierarchy. In each inductive step, the proof is by contradiction.

The base case for the induction is a topology with a one-level hierarchy ($k = 1$). The proof for LVA assumes that an infinite number of updates (*add* of *delete*) is generated for some link, and it is shown that this is impossible due to the finite number of nodes in the network and the fact that the node detecting the link change sends exactly one update. Because flat LVA is used within the lowest-level areas and no information is propagated outside an area concerning topology changes within the area, it is clear that the algorithm terminates for such changes.

It remains to show that the algorithm terminates if there is a change in the connectivity between two areas. Note that all nodes use LVA on the graph comprising all inter-area links in the network. The exact same argument used for flat LVA can now be used: assume that an infinite number of updates is generated. This implies that there is at least one node that generates an infinite sequence of updates about some link l. In turn, this implies that a neighbor of this node also generates an infinite sequence of updates concerning the same link. The proof for LVA proceeds showing that there must be an infinite sequence of nodes who start sending infinitely many updates caused by that same change. However, this argument is valid only if nodes can validate updates, which can be accomplished by sequence numbers. Here lies the only difference in the proof. Because those links can be detected at multiple heads-of-link, it must be assured that this does not lead to a "flip-flop-effect," where nodes switch between the heads-of-link whose sequence number they use for reporting the link. This problem does not apply to the two areas that are actually connected, in these areas the links are reported with their physical heads-of-link, inclusion of these links does not alter the termination property of LVA within areas. In remote areas, if we require that all nodes who receive multiple heads choose the head that they propagate using the same criteria (for example, using the smallest address,) then no infinite sequence of adds and deletes will be created. Hence, ALVA terminates when there is one level of hierarchy.

Now consider a topology with $k > 1$ levels of hierarchy. Assume that ALVA terminates for $k-1$ levels of hierarchy. A k-level hierarchical topology is composed of $(k-1)$-level areas and links connecting these areas. By the inductive hypothesis, we know that ALVA terminates for any changes within the $(k-1)$-level areas. It remains to show that ALVA terminates if there is a change in the connectivity between two k-level areas. The argument here is very similar to the case of one level of hierarchy. Again, LVA is used at the k-th level of hierarchy, and the only difference to the flat case is that there can be multiple heads-of-link, that could cause the described "flip-flop-effect." Again, this is prevented by requiring the consistent choice of the head-of-link that is reported to a neighbor. Hence, ALVA terminates in a k-level hierarchy. **q.e.d.**

Lemma 2: Within finite amount of time after t_0, all routers have the consistent information necessary to compute correct source graphs.

Proof: The proof that all nodes have consistent information when ALVA terminates is by induction on the levels in the hierarchy.

Again, the base case for the induction is a one-level hierarchy.

The proof that information is consistent in a finite time after topology changes cease for LVA is by induction over the length of paths in hops. In a similar fashion, we can argue the same case in the higher level.

Within any area, LVA is used. Therefore, a any node – in particular any border node – in the area has correct information about the topology within the area that it is part of. In addition, it knows about all links that it needs to route to neighboring areas. (A neighboring area appears as a destination in the level 0 topology table, therefore, every node – including border-nodes – knows at least one link to that destination). Because a border node forwards the latter information to its neighbors in the other nodes, the border-nodes of these areas know about the connectivity of their neighboring area at level 1. This is the base case, one hierarchical hop. The information is propagated within the neighbor-area, using LVA rules for a network at level 1, where minimum hop routing is used Xfor the computation of the preferred paths. From the correctness of Xthis (flat) LVA at the higher level follows the correctness of the hierarchical scheme.

The formal proof for this argument uses induction, as in the flat case. The base case – neighboring areas – is described above. Then, for $h > 1$ hops assume that the correct, needed information to reach a destination is known in areas that are less than h hierarchical hops away from this destination. Consider a path that spans h hierarchical hops. Then, we know that there is a flat path to the first area on that path. This area is $h-1$ hierarchical hops away from the destination. The subpath from that area to the destination must be optimal; therefore, by the inductive hypothesis, it must be known in that area. However, because it is used and known, it must be propagated to all the neighboring areas by its border nodes. It then follows that it must also be known in the area we first considered. This, together with the known path to that neighboring area, means that the complete path is known h hierarchical hops away.

Now consider the case that we have $k > 1$ levels of hierarchy. Assume that the algorithm yields consistent information for $k-1$ levels. A k-level area is composed of $(k-1)$-level areas. By the inductive hypothesis, all nodes have consistent information about their $(k-1)$-level. In addition, the links between $(k-1)$-areas, as well as their connectivity to outside areas is known to all $(k-1)$-level border-nodes. Hence, for a given k-level area, this information is also known at all its border-nodes, since a k-level border-node is also a border-node at levels $1, \ldots k$. Then, we can use the same inductive argument as in the base case, using the links between k-level areas as hierarchical hops.

This proves Lemma 2. **q.e.d.**

V. Performance

Given that LVA has been shown to outperform the ideal link-state algorithm in [5], it can be expected that ALVA performs better than area-based schemes based on flooding, such as OSPF, by reducing the control traffic both within areas as well as across the backbone. To verify this expectation, we compared ALVA with OSPF in several simulations. Simulations were performed using random graphs with 100 nodes. Nodes had an average degree of approximately 3. Recent work [16] shows that this is a realistic node degree for internetworks. The topologies were produced according to two general schemes. According to the first scheme, there is a backbone with 56 nodes, one area with 30 nodes, and 14 stub areas with one node each. We chose to use stub areas because we were particularly interested in the effect of changes in the backbone. This topology type allows us to have many destination areas but to focus on the effects that a change in the backbone has within the backbone and in the complete area. In the second scheme, the backbone contains 40 nodes and there are four areas with 15 nodes each.

To obtain random topologies according to these schemes, for each area (including the backbone) nodes are places randomly in a plane. Any two nodes u, v within the area are then connected according to the exponential model as proposed by Zegura et al. [17]. In addition, we make sure that all areas are connected graphs. Then, each area is connected with the backbone at two randomly chosen nodes. This method to obtain topologies allows us to study networks that exhibit the characteristic of the logical star configuration that OSPF requires for inter-area traffic [4].

To simulate OSPF, we make the following assumptions:

- Areas contain exactly one mask, i.e., they are seen as a single entity from outside the area. In terms of storage and communication overhead needed, this actually presents the best case for OSPF.
- Border nodes belong to exactly one area and the backbone. A border node runs two copies of the flooding algorithm, one for the backbone and one for the area to which it belongs.
- A border node reports to the backbone that it has a link to the area of which it is part.
- A border node reports within its area links to all other areas with costs as determined by the shortest-path algorithm in the backbone topology.

We evaluate the performance in terms of update messages sent and number of steps required for the algorithms to converge. When a node receives an update, it compares its local step counter with the sender's, takes the maximum, and increments the counter. This way we obtain the number of sequential update message exchanges between neighbors needed for convergence. In addition, we compare the size of the topology tables stored by the routers. In terms of these criteria, the assumptions stated above actually represent the best case for OSPF. For our simulation, we assume that control packets are transmitted error free and are processed one at a time in the order received. OSPF and protocols based on ALVA provide their own retransmissions. Using equivalent mechanisms, ALVA requires less overhead than topology-broadcast to ensure reliable transmission of updates. Packets sent over failed links are dropped. To detect new connectivity or link failures, a simple hello protocol was used (much like in the OSPF specification).

Figures 10 through 14 show the results of our simulations. Results are shown for changes in link cost, link failures, link establishments, node failures and node establishments. The bars represent the average (mean) number of messages and steps, respectively, while the markers show the standard deviation. To obtain these results, we performed the changes for every single link and node in the network. Thus, no sampling errors need to be presented.

Figures 10 and 11 shows the overall results for one representative topology of each class. These results include changes at the borders as well as changes in the backbone and other areas.

In Figures 12 and 13, more detailed results are shown. The left graph in figure 12 shows the message sent for changes within the backbone, while the right graph in that figure represents changes within the other area of the topology of the first type. Similarly, Figure 13 shows the number of messages until convergence for changes within the backbone and in one of the areas for the topology of the second type.

It is clear that ALVA needs less time and fewer messages to converge for changes in single links in the backbone as well as the areas. OSPF behaves better only when nodes fail. As expected, the number of messages required when links change or are established within an area for OSPF is constant. (This is not true for link failures, because some of the failures may disconnect nodes or partition the graph). The deviation from the mean for such changes in the backbone shows that changes in the backbone cause traffic in the areas in addition to the traffic within the backbone.

In all simulations, ALVA clearly outperforms OSPF when link changes occur, links fail, or new links are established. The only case where OSPF converges with less overhead is when a node fails. In this case, the delete operation in LVA causes ALVA to create slightly more packets than OSPF.

Figure 14 shows the average size of the topology tables at routers in the backbone and in another area for both types of topologies. It can be seen that routers using ALVA need to keep only about half as many links in their tables on the average when compared to routers OSPF. This is true in particular for backbone routers, which include the border-nodes that run two copies of the topology broadcast (one for the backbone and one for their area). As the size of the areas grows, this advantage for ALVA becomes even more pronounced. We have also obtained results using flat LVA for larger topologies than the areas shown here. The results

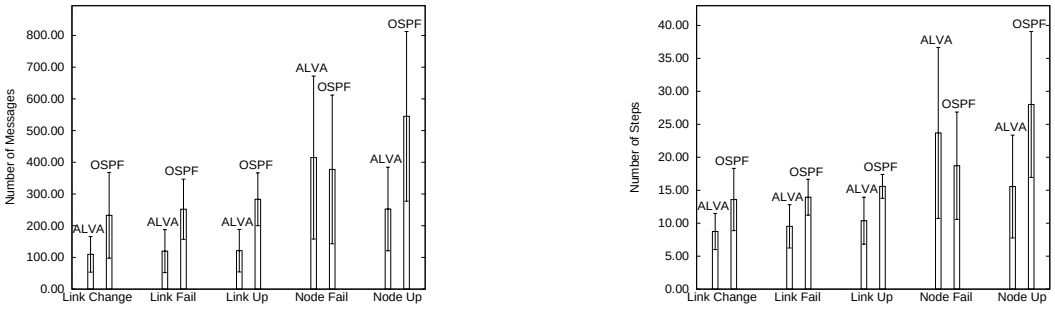

Fig. 10. Topology type 1

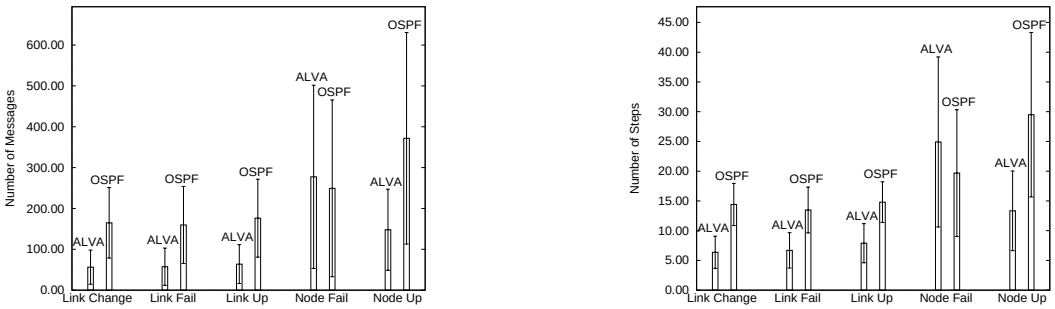

Fig. 11. Topology type 2

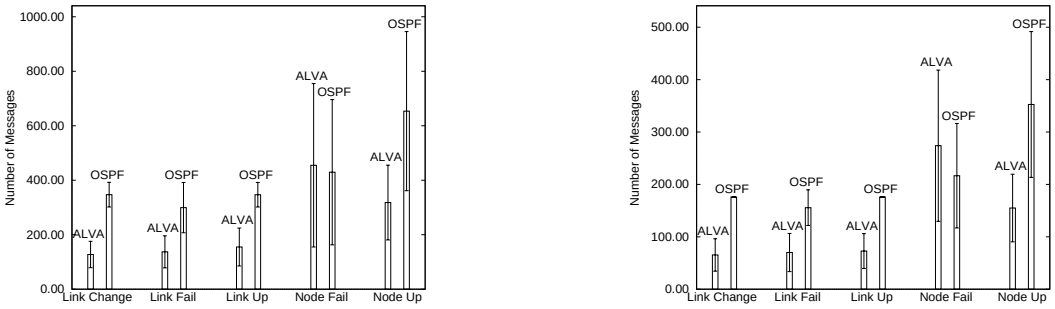

Fig. 12. Topology type 1, backbone (left) and area (right)

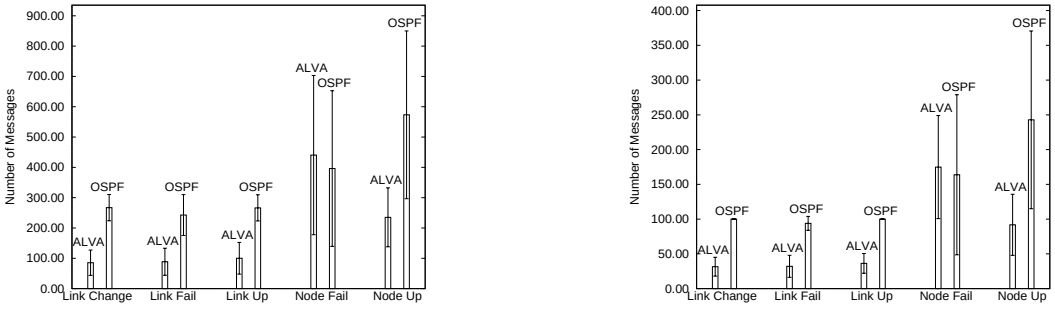

Fig. 13. Topology type 2, backbone (left) and area (right)

obtained that way confirmed the significant savings in the number of links in the tables.

As an internetwork grows larger, the backbone-based topology required in OSPF forces the backbone to become larger or the areas that connect to it to grow larger. Figure 15 illustrates the savings that can be derived with ALVA over OSPF by not requiring a backbone. The topology used for this experiment is of type 1. There are 179 edges in the topology, giving the nodes an average degree of 3.58, with a maximum degree of 9. For the first part of the experiment, the node addresses were chosen such that the backbone was partitioned into three connected areas; ALVA was used in this scenario. For the second part, the backbone was one contiguous area; both ALVA and OSPF were used in this scenario. The results of this experiment show that the more flexible choice of topologies can widen the margin by which ALVA outperforms OSPF significantly. With the large backbone area required by OSPF partitioned into smaller areas, ALVA outperforms OSPF even when nodes fail.

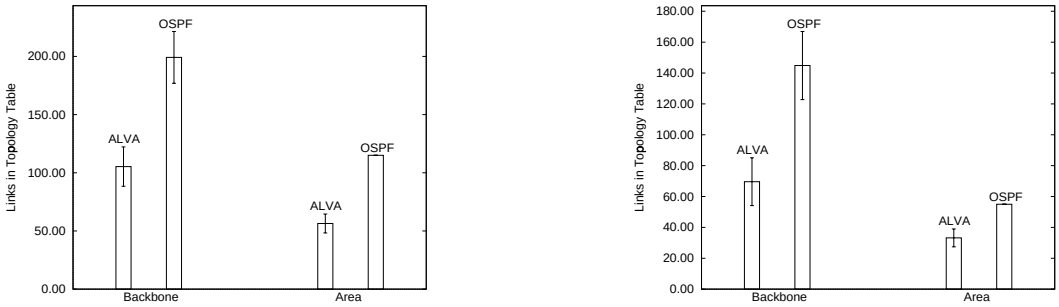

Fig. 14. Size of topology tables. left: topology type 1; right: topology type 2

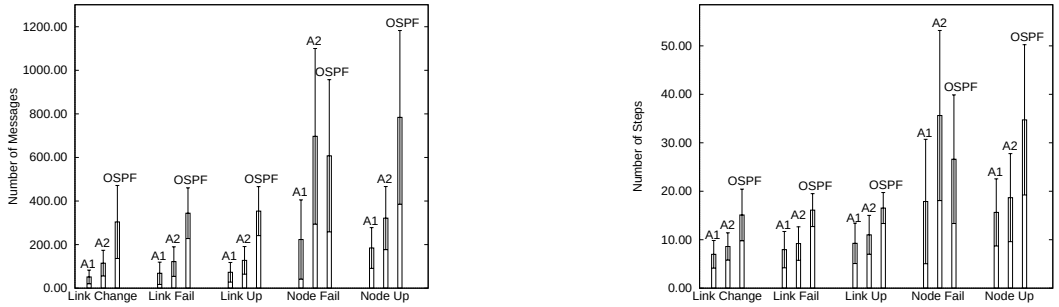

Fig. 15. Topology of type 1 with backbone that can be partitioned (A1: ALVA with backbone area partitioned into three areas; A2: ALVA with contiguous backbone area; OSPF with contiguous backbone area)

In addition, in contrast to OSPF, ALVA allows for multiple levels of hierarchy and ALVA does not require a backbone, which means that very large backbones can be broken into smaller areas that provide the same connectivity. These added features make it possible to further reduce communication as well as storage overhead.

VI. Conclusions

We have presented a new hierarchical routing algorithm based on link-vector routing and areas for aggregation of routing information. The main idea of LVA is to use link-state information to compute optimal paths but without replicating the complete topology information at every node. This idea has been extended to allow multiple levels of hierarchy. At each level of the hierarchy, partial topology is stored.

The performance of ALVA was compared with that of OSPF. Our simulation results show that, even with only one level of hierarchy, ALVA clearly outperforms OSPF in terms of storage and communication requirements. ALVA does not require a backbone-centered topology, and our simulation experiments illustrate performance advantages gained by allowing arbitrary area-based topologies. In addition, allowing multiple levels of hierarchy makes the new algorithm far more scalable than OSPF. ALVA constitutes the basis for the development of more efficient Internet routing protocols based on link-state information.

References

[1] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, Englewood Cliffs, NJ, 2nd edition, 1992.

[2] International Standards Organization, *Intra-Domain IS-IS Routing Protocol*, ISO/IEC JTC1/SC6 WG2 N323, Sept. 1989.

[3] D. Estrin, M. Steenstrup, and G. Tsudik, "A protocol for route establishment and packet forwarding across multidomain internets," *IEEE/ACM Transactions on Networking*, vol. 1, no. 1, pp. 56–70, February 1993.

[4] J. Moy, "OSPF Version 2," RFC 1583, Network Working Group, March 1994.

[5] J.J. Garcia-Luna-Aceves and J. Behrens, "Distributed, scalable routing based on vectors of link states," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1383–95, October 1995.

[6] F. Kamoun, *Design Considerations for Large Computer Communication Networks*, Ph.D. thesis, University of California, Los Angeles, 1976.

[7] M. Steenstrup, *Routing in Communications Networks*, Prentice Hall, Englewood, Cliffs, NJ, 1995.

[8] J. McQuillan, "Adaptive routing algorithms for distributed computer networks," BBN Rep. 2831, Bolt Beranek and Newman Inc., Cambridge MA, May 1974.

[9] L. Kleinrock and F. Kamoun, "Hierarchical routing for large networks: Performance evaluation and optimization," *Computer Networks*, vol. 1, no. 3, pp. 155–174, 1977.

[10] P.F. Tsuchiya, "The landmark hierarchy: A new hierarchy for routing in very large networks," *Computer Communications Review*, vol. 18, no. 4, pp. 43–54, 1988.

[11] C. Alaettinoglu and A.U. Shankar, "The viewserver hierarchy for interdomain routing: Protocols and evaluation," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1396–1410, October 1995.

[12] S. Murthy and J.J. Garcia-Luna-Aceves, "Loop-free internet routing using hierarchical routing trees," in *Proc. IEEE INFOCOM 97*, Kobe, Japan, April 7–11 1997.

[13] J.J. Garcia-Luna-Aceves and S. Murthy, "A path finding algorithm for loop-free routing," *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, pp. 148–160, February 1997.

[14] C.V. Ramamoorthy and W.T. Tsai, "An adaptive hierarchical routing algorithm," in *Proc. COMPSAC 83*, Chicago, IL, USA, November 1983, IEEE, pp. 93–104.

[15] W.T. Tsai, C.V. Ramamoorthy, W.K. Tsai, and O. Nishiguchi, "An adaptive hierarchical routing protocol," *IEEE Transactions on Computers*, vol. 38, no. 8, pp. 1059–1075, Augst 1989.

[16] R. Govindan and A. Reddy, "An analysis of internet inter-domain topology and route stability," in *Proc. IEEE INFOCOM '97*, Kobe, Japan, April 7-11 1997.

[17] E.W. Zegura, K.L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings IEEE INFOCOM'96*, San Francisco, CA, March 26–28 1996, vol. 2, pp. 594–602.